POEM: Pattern-Oriented Explanations of CNN Models

Vargha Dadvar
University of Waterloo
vdadvar@uwaterloo.ca

Lukasz Golab
University of Waterloo
lgolab@uwaterloo.ca

Divesh Srivastava
AT&T Chief Data Office
divesh@research.att.com

ABSTRACT

Deep learning models achieve state-of-the-art performance in many applications, but their prediction decisions are difficult to explain. Various solutions exist in the area of explainable AI, for example to understand individual predictions or to approximate complex models using simpler interpretable ones. We contribute to this body of work with POEM: a tool that produces pattern-oriented explanations of image classification models. POEM explains models that learn hierarchies of concepts, such as Convolutional Neural Networks that detect shapes and objects in images. For example, POEM may identify a pattern of the form "if bed then bedroom", indicating that if an image contains a bed and the model pays attention to this region of the image during inference, then the model classifies the image as a bedroom. We present the modular design of POEM, followed by examples of POEM's use in model auditing and detecting errors in training data.

PVLDB Reference Format:
Vargha Dadvar, Lukasz Golab, and Divesh Srivastava. POEM: Pattern-Oriented Explanations of CNN Models. PVLDB, 15(12): 3618-3621, 2022.
doi:10.14778/3554821.3554858

1 INTRODUCTION

Deep learning models achieve state-of-the-art performance in many applications, including computer vision and natural language processing. However, these models are complex and usually do not give human-interpretable explanations of their decisions. The lack of explainability may prevent users from trusting the models. This creates a critical barrier to adoption, especially in high-stakes applications such as healthcare and in places such as the European Union where algorithmic explainability is required by law.

Solutions to this problem include methods to explain individual predictions, for example, by quantifying the importance of the features to the model's decision [14, 15]. Another approach is to use *surrogate models*, which are interpretable models such as decision trees that can approximate black-box models [4, 6, 10]. As our contribution towards explainable AI, we present POEM: a tool that produces pattern-oriented explanations of image classification models.

We focus on models that learn hierarchies of concepts. One example is a Convolutional Neural Network (CNN), which is commonly used in computer vision. A CNN consists of multiple layers that detect shapes, textures and objects in images. For example, suppose we wish to explain a given CNN that classifies images of rooms into kitchens and bedrooms. The goal of POEM is to associate the detected concepts with the model's prediction decisions. POEM outputs patterns of concepts in the form of rules. For instance, POEM may identify a pattern of the form "if bed then bedroom", indicating that if an image contains a bed and the model pays attention to this region of the image during inference, then the model classifies the image as a bedroom rather than a kitchen.

We give an overview of CNNs in Section 2. Then, in Section 3, we present the modular architecture of POEM. The first step, "Concept Identification", finds the concepts learned by the given model. In the second step, "Concept Attribution", POEM locates these concepts in the input images and associates the concepts with prediction decisions. The third step, "Concept Pattern Mining" produces interpretable rules linking concepts to prediction decisions. As we will explain in Section 3, the novelty of POEM is in the use of state-of-the-art components in the first two steps and the use of new rule mining and visualization methods (that have not yet been applied to explain image classification models) in the third step.

In Section 4, we demonstrate POEM using a sample CNN model and dataset. We show how the concept patterns produced by POEM enable model auditing. POEM also allows users to visualize the concepts learned, display images that were classified correctly or incorrectly, and display images that are exceptions to the discovered concept patterns. Furthermore, we show how concept patterns can serve as data quality assessment tools to identify mislabelled images in training data (e.g., those labelled as bedrooms that should be labelled as kitchens).

2 CONVOLUTIONAL NEURAL NETWORKS

We begin with an overview of CNNs for image classification. For black-and-white images, the input consists of a two-dimensional matrix of pixel intensities in the image. For colour images, the input consists of three such matrices, corresponding to the pixel intensities of the three primary colours. The output consists of the most likely class label for the given image.

To make a prediction, the input features pass through multiple layers in the network. Some of these layers are *convolutional*, whose purpose is to transform the input pixels in a way that can detect shapes and objects. These transformations are done using *filters*, and the output of a filter is called an *activation map*.

An activation map is a two-dimensional numeric array indicating the locations within a given image that were activated by the filter. For example, a simple filter may identify pixels in an image that correspond to straight lines. We say that an activation map is *highly activated* if at least one of its elements has a high value (above some threshold, such as the 99-percentile of the filter's activation values over all the images in the given dataset).

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554858

Training a CNN is done by passing the training images through the model, and updating the filters by *backpropagating* the prediction errors. In this way, the filters learn to identify shapes and objects in images that reduce the prediction error of the model.

The first few convolutional layers may identify simple patterns such as edges and corners, and the subsequent layers are more likely to detect more complex shapes. In POEM, as in previous work on explaning CNNs [1, 2, 13, 14], we focus on the last convolutional layer. This is where the network is likely to detect human-understandable concepts in images.

3 SYSTEM OVERVIEW

POEM consists of three modules illustrated in Figure 1 and described in detail below. Our current implementation of POEM is geared towards CNNs; in future work, we will study extensions to other network architectures in which it is possible to identify concepts and attribute concepts to prediction decisions.

3.1 Concept Identification

In this step, the goal is to identify the concepts learned by the model. In a CNN, this means identifying the shapes and objects detected by the filters in the last convolutional layer.

There exist several methods to identify concepts in CNNs [1, 3, 5, 8, 11, 18]. In POEM, we use the latest version of a method called *Network Dissection* [2]. This method first segments each image in the given dataset into concepts present in the image, such as objects, object parts, textures, and colours. This is done using a semantic segmentation model called *Unified Perceptual Parsing* [17], which is pretrained to identify a wide range of such concepts. We then pass each image through the CNN, and we measure the pixel overlap between these concepts and the filter activation maps. This process maps a filter to the most likely concept it is detecting. We show three examples in Figure 1, with beds, stoves and headboards, respectively, mapped to three filters.

This concept identification process can also be applied to CNN models used in specialized domains such as medical image analysis, by pretraining the segmentation model to identify concepts that are meaningful in the target domain.

Among the related work, *ERIC* [16] requires manual inspection of activation images to identify filter-concept mappings, which is not scalable. *ACDTE* [4] outputs unlabeled clusters of activated regions, which are not guaranteed to correspond to concepts. A system that more closely resembles our pipeline is *CNN2DT* [10], but it uses an earlier network dissection method [1] that relies on a separate proxy dataset with pre-labeled concepts. As a result, the concepts identified by *CNN2DT* may not exist in the dataset used when explaining the model.

3.2 Concept Attribution

For each image, the goal of this step is to attribute the most likely concepts that played a role in the model's prediction decision. For this, the concept must exist in the image. Additionally, we want to ensure that the network paid attention to this concept when making the prediction.

More precisely, we attribute a concept to an image if: 1) the concept is present in the image; 2) a filter mapped to this concept (in the Concept Identification step described above) is highly activated when the image is passed through it; and 3) there is significant overlap between the location of the concept in the image and the highly activated area in the related filter activation map.

Related approaches such as *ERIC* and *CNN2DT* do not check conditions 1 and 3 above, leading to false positives compared to our method. The semantic segmentation model mentioned earlier allows us to locate concepts in images and check these additional conditions. To check the concept-activation overlap, we first resize each filter activation map to the size of the input image, and we superimpose the map onto the image. We then check that at least 50% of the highly activated area is covered by the concept found in the image. Finally, we discard the weakest concepts, which are those activated in less than 1 percent of the images in the given dataset.

The output of this step is a transformed dataset of images, represented by their important concepts as features, and the CNN model predictions as the class label. We show three examples in Figure 1.

3.3 Concept Pattern Mining

To find patterns relating concepts to model predictions, such as the examples in Figure 1, we apply an ensemble of rule mining methods on the concepts produced in the Concept Attribution step.

Related work [4, 10, 16, 18] uses decision trees and the rules extracted from them (i.e., the root-to-leaf paths) to explain the relationships between concepts and predictions. In POEM, in addition to CART (Classification and Regression Trees), we support two recent rule mining methods: *Explanation Tables* [7] and *Interpretable Decision Sets (IDS)* [12].

Explanation tables consist of a set of patterns that together maximize the information about the distribution of the class label. Unlike decision trees, patterns from explanation tables can overlap and are geared toward informative and concise data explanation rather than out-of-sample predictive power.

IDS finds a set of rules by balancing several optimization criteria such as support, confidence and conciseness. IDS was designed to be an interpretable classifier, and we use each rule independently for explanation.

3.4 Web Interface

Figure 2 shows POEM's Web interface. We explain the interface features below, and in Section 4 we will show how to use POEM to explain CNNs.

There are three panels: *settings* on the left, *patterns* at the top, and *images* at the bottom. A user begins a session in the settings panel, by selecting a dataset, a CNN model, and the desired rule mining methods and their settings. These settings are prepopulated with default values and include the minimum support threshold for each method. For CART, the minimum support corresponds to the minimum fraction of examples in each leaf of the tree. These parameters indirectly control the total number of patterns to find using each method.

After pressing the "Compute Patterns" button, the patterns panel shows the output. Each rule consists of the concepts, the corresponding prediction made by the selected model, a support fraction (what fraction of the examples in the selected dataset match the concepts

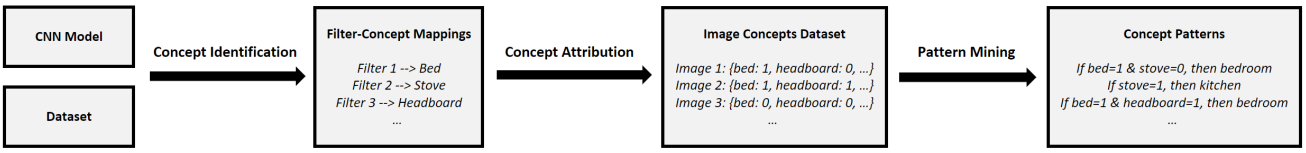

Figure 1: Overview of POEM

in the rule), a confidence fraction (how many examples having the concepts mentioned in the rule are predicted as shown in the rule), an accuracy fraction (how many examples having the concepts mentioned in the rule and predicted as in the rule are predicted correctly based on the ground-truth labels), a score (details below), the method that produced the rule, and a set of options for further analysis. For example in Figure 2, pattern 1, found by IDS, states that in 19% of the images (as indicated by support), the presence of a bed results in the model predicting a bedroom rather than a kitchen or living-room. Furthermore, as indicated by a confidence of 99%, this combination of concepts results in a prediction of a bedroom 99% of the time, and 99% of such examples are predicted correctly as bedrooms by the model, as shown by the accuracy.

By default, the rules are ordered using a score, computed as $\frac{\text{support} \times \text{confidence}}{\text{rule size}}$, where rule size refers to the number of concepts. This sort order highlights concise and confident patterns that cover a large subset of the data. However, the user can also sort the rules by support, confidence, accuracy or the method.

In contrast to similar tools such as *CNN2DT*, which only visualize the top activated images for each concept, POEM allows users to view different categories of images related to each pattern to extract further insights about the CNN model and the data. For this purpose, we can select a pattern from the list, and then choose one of the coloured buttons that appear, as shown for pattern 1 in Figure 2. The following options are available for each pattern:

- *Matching* images (green button) match both the concepts and the prediction shown in the given rule. For example for pattern 1 in Figure 2, matching images are those attributed to the concept 'bed' and predicted as bedrooms.
- *Non-matching* images (yellow button) match the concepts mentioned in the given rule but not the prediction. For pattern 1 in Figure 2, non-matching images are those attributed to the concept 'bed' but predicted as either kitchens or living rooms. This happens when confidence is less than 100%, indicating that the model's predictions were sometimes different than the label stated in the rule.
- *Wrongly-predicted* images (red button) match the concepts and the model's prediction shown in the given rule, but have a different label in the dataset. For pattern 1 in Figure 2, wrongly-predicted images are those which are attributed to the concept 'bed', were predicted as bedrooms, but are labelled in the given dataset as kitchens or living rooms.

Choosing one of the above options displays the requested images in the images panel, as shown at the bottom of Figure 2. In the images panel, it is also possible to choose a specific concept from a selected rule, which displays the images with the related filter's activation areas highlighted on the image. For example, in Figure 2, the images matching pattern 1 are displayed, with their bed concept activations highlighted. This allows us to check the correspondence between the concepts in images and the highly activated areas of the CNN filters.

3.5 Implementation

We implemented the POEM backend using Python and its scientific libraries, while the web application is based on JavaScript libraries VueJS as the frontend and NodeJS as the backend. For network dissection, we used the code from the project's Github page[1] with some modifications. We also used the Github code[2] for unified perceptual parsing as the semantic segmentation model. For explanation tables, we obtained the code from the authors, while for IDS, we used the Github code[3], and for CART we used the implementation from the Scikit-learn package.

4 EXAMPLE: RECOGNIZING BEDROOMS, KITCHENS AND LIVING ROOMS

In this section, we describe one example of using POEM to explain an image recognition model. In this example, we analyze the concepts learned in the last convolutional layer of the *ResNet-18* CNN model [9] for the task of indoor place classification. We use the *Places* dataset [19], which includes images from 365 classes. For pattern analysis, it makes sense to focus on a few classes to be compared against each other, which is why we only consider bedrooms, kitchens and living rooms in this example. The Places dataset contains 5000 images for each of these classes.

We use a ResNet-18 model pretrained on the entire Places dataset, but we replace the output layer of the model to match the three target classes. We then fine-tune the model on the bedroom, kitchen and living room images, with only the output layer being trained, and the rest of the network being used as a fixed feature extractor. The resulting model has a 92.4% prediction accuracy in distinguishing between the images of the three selected classes.

Figure 2 shows the patterns. We only show patterns that indicate the presence of concepts. We see that concepts such as bed in bedrooms, work surface in kitchens, and sofa in living rooms lead to almost certain predictions by the model. This indicates that the model is mostly using the right concepts for its decisions.

Notably, these patterns do not just reflect the presence of concepts in images, which could lead to a data correlation trap, but also indicate that the model is likely to be looking at these concepts

[1] https://github.com/davidbau/dissect
[2] https://github.com/CSAILVision/unifiedparsing
[3] https://github.com/lvhimabindu/interpretable_decision_sets

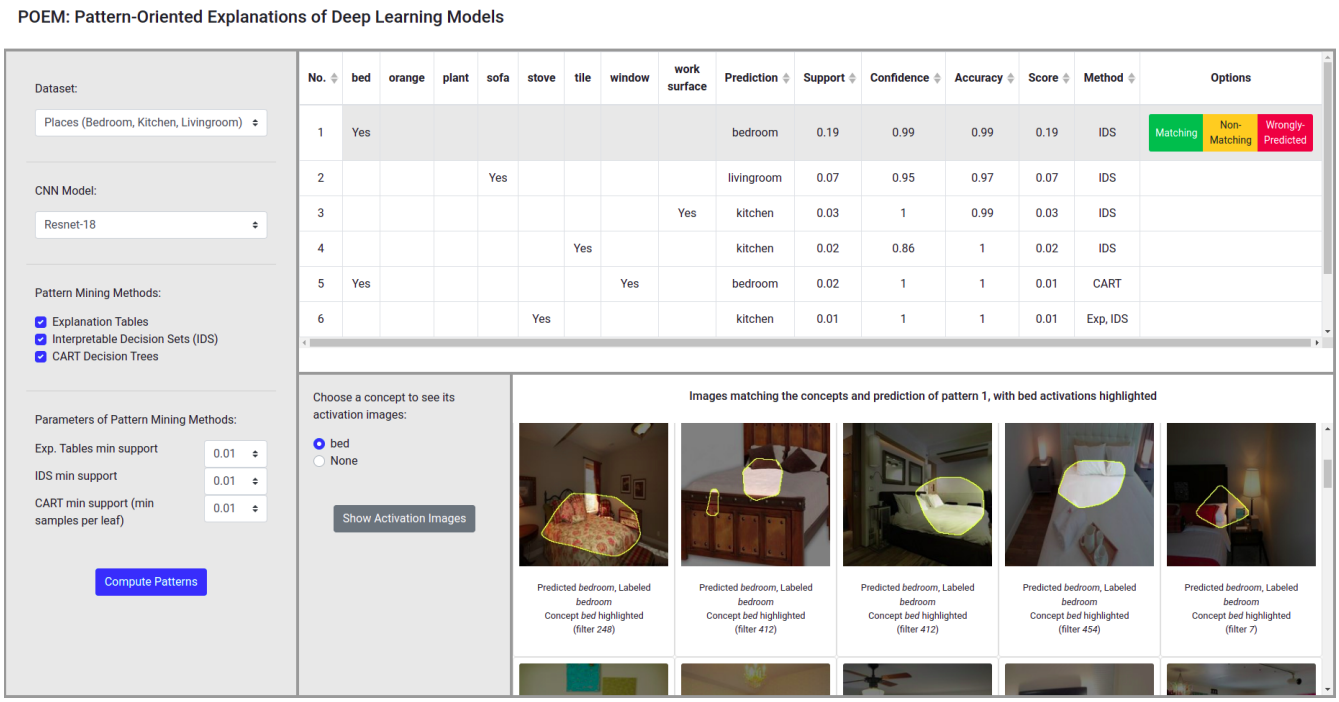

Figure 2: POEM interface showing the rules for bedroom vs. kitchen vs. living room

in images. This is because we check multiple criteria to attribute concepts to images to make sure the CNN filters pay attention to the concepts, as explained in Section 3.2.

To confirm that the concepts in images correspond to the related highly activated areas in the model, we can view the matching images for each pattern. In the bottom panel of Figure 2, we see that beds are present in the images matching pattern 1, and the filter activations overlap with either an entire bed or part of a bed.

We can then examine patterns that do not have 100% confidence or 100% accuracy. For example, we may want to know why images attributed to a tile in pattern 4 were not predicted as kitchens in 14% of the cases. We select the non-matching button for pattern 4 to see examples of such images. By examining these images, we can find out whether the dataset includes images of bedrooms or living rooms with tiles, or images mislabeled as bedrooms or living rooms. We may also find examples that are more complex for the model to predict, possibly because of a lack of distinguishing concepts.

REFERENCES

[1] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. 2017. Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *Computer Vision and Pattern Recognition*.

[2] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences* (2020).

[3] Z. Chen, Y. Bei, and C. Rudin. 2020. Concept whitening for interpretable image recognition. *Nature Machine Intelligence* 2 (12 2020), 1–11.

[4] R. El Shawi, Y. Sherif, and S. Sakr. 2021. Towards Automated Concept-based Decision Tree Explanations for CNNs. In *EDBT 2021 24th International Conference on Extending Database Technology*.

[5] R. Fong and A. Vedaldi. 2018. Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks. *arXiv preprint arXiv:1801.03454* (2018).

[6] N. Frosst and G. E. Hinton. 2017. Distilling a Neural Network Into a Soft Decision Tree. *ArXiv* abs/1711.09784 (2017).

[7] K. El Gebaly, G. Feng, L. Golab, F. Korn, and D. Srivastava. 2018. Explanation Tables. *IEEE Data Engineering Bulletin* 41 (2018), 43–51.

[8] A. Ghorbani, J. Wexler, J. Zou, and B. Kim. 2019. Towards Automatic Concept-based Explanations. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[9] K. He, X. Zhang, S. Ren, and J. Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:arXiv:1512.03385

[10] S. Jia, P. Lin, Z. Li, J. Zhang, and S. Liu. 2020. Visualizing Surrogate Decision Trees of Convolutional Neural Networks. *J. Vis.* 23, 1 (feb 2020), 141–156.

[11] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viégas, and R. Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *ICML*.

[12] H. Lakkaraju, S. Bach, and J. Leskovec. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1675–1684.

[13] A. Mahendran and A. Vedaldi. 2016. Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images. *International Journal of Computer Vision* 120 (12 2016).

[14] R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 618–626.

[15] K. Simonyan, A. Vedaldi, and A. Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:arXiv:1312.6034

[16] J. Townsend, T. Kasioumis, and H. Inakoshi. 2021. ERIC: Extracting Relations Inferred from Convolutions. In *Computer Vision – ACCV 2020*. Springer International Publishing, 206–222.

[17] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. 2018. Unified Perceptual Parsing for Scene Understanding. In *European Conference on Computer Vision*. Springer.

[18] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu. 2018. Interpreting CNNs via Decision Trees. arXiv:arXiv:1802.00121

[19] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. 2017. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).